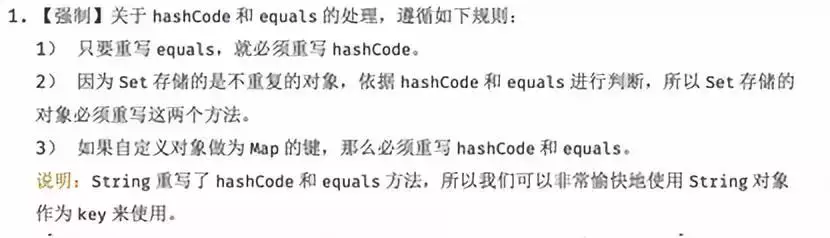
<https://mp.weixin.qq.com/s?__biz=MzIyNDU2ODA4OQ==&mid=2247483944&idx=1&sn=3a81af53aaaac56b60e50c1619433513&chksm=e80db45edf7a3d48807151107efd8d2b39f04424b6b1fe57212a92ea2b4927702a5250c37fa7&scene=21#wechat_redirect>

# 说说hashCode() 和 equals() 之间的关系？

先祭一张图，可以思考一下为什么？



## ****介绍****

equals() 的作用是用来判断两个对象是否相等。

hashCode() 的作用是获取哈希码，也称为散列码；它实际上是返回一个int整数。这个哈希码的作用是确定该对象在哈希表中的索引位置。

## ****关系****

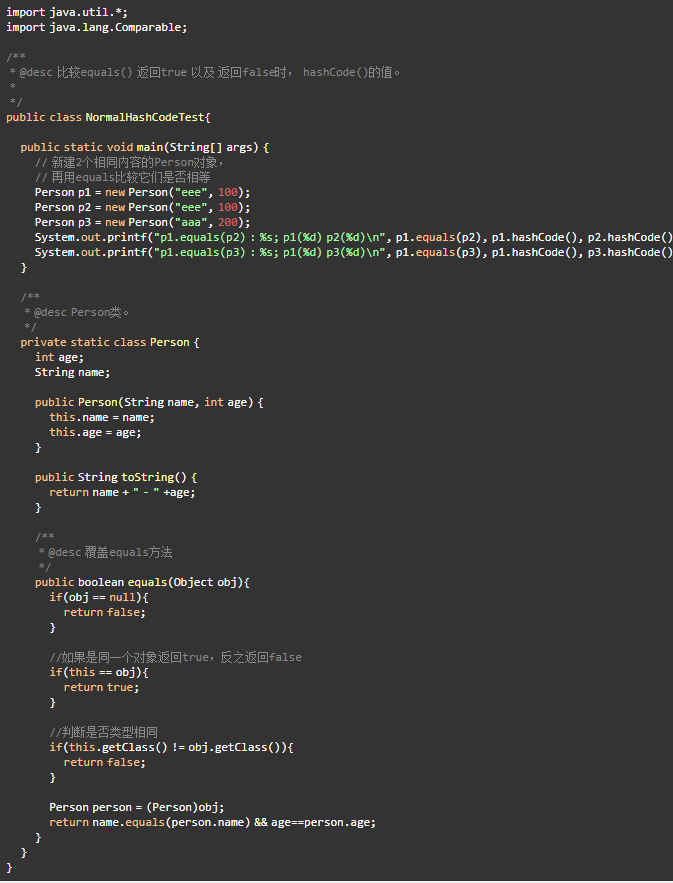
我们以“类的用途”来将“hashCode() 和 equals()的关系”分2种情况来说明。

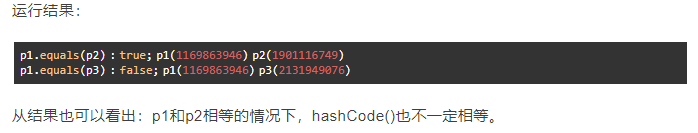
### 1、不会创建“类对应的散列表”

这里所说的“不会创建类对应的散列表”是说：我们不会在HashSet, Hashtable, HashMap等等这些本质是散列表的数据结构中，用到该类。例如，不会创建该类的HashSet集合。

**在这种情况下，该类的“hashCode() 和 equals() ”没有半毛钱关系的！**equals() 用来比较该类的两个对象是否相等。而hashCode() 则根本没有任何作用。

下面，我们通过示例查看类的两个对象相等 以及 不等时hashCode()的取值。





### 2、会创建“类对应的散列表”

这里所说的“会创建类对应的散列表”是说：我们会在HashSet, Hashtable, HashMap等等这些本质是散列表的数据结构中，用到该类。例如，会创建该类的HashSet集合。

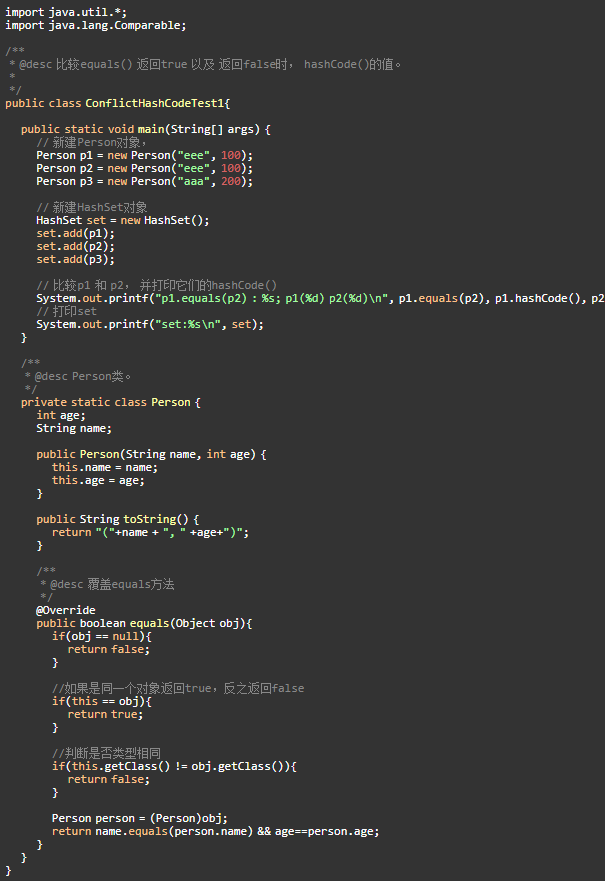
在这种情况下，该类的“hashCode() 和 equals() ”是有关系的：

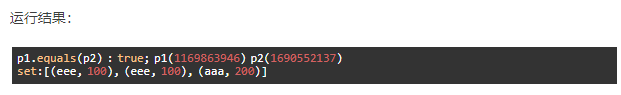
* **如果两个对象相等，那么它们的hashCode()值一定相同。**这里的相等是指，通过equals()比较两个对象时返回true。
* **如果两个对象hashCode()相等，它们并不一定相等**。因为在散列表中，hashCode()相等，即两个键值对的哈希值相等。然而哈希值相等，并不一定能得出键值对相等。补充说一句：“两个不同的键值对，哈希值相等”，这就是哈希冲突。

此外，在这种情况下。若要判断两个对象是否相等，除了要覆盖equals()之外，也要覆盖hashCode()函数。否则，equals()无效。

举例，创建Person类的HashSet集合，必须同时覆盖Person类的equals() 和 hashCode()方法。

如果单单只是覆盖equals()方法。我们会发现，equals()方法没有达到我们想要的效果。





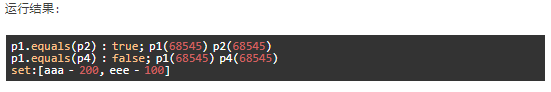
结果分析：

我们重写了Person的equals()。但是，很奇怪的发现：HashSet中仍然有重复元素：p1 和 p2。为什么会出现这种情况呢？

**这是因为虽然p1 和 p2的内容相等，但是它们的hashCode()不等；所以，HashSet在添加p1和p2的时候，认为它们不相等。**

那同时覆盖equals() 和 hashCode()方法呢？





结果分析：

这下，equals()生效了，HashSet中没有重复元素。

比较p1和p2，我们发现：它们的hashCode()相等，通过equals()比较它们也返回true。所以，p1和p2被视为相等。

比较p1和p4，我们发现：虽然它们的hashCode()相等；但是，通过equals()比较它们返回false。所以，p1和p4被视为不相等。

## ****原则****

**1.同一个对象（没有发生过修改）无论何时调用hashCode()得到的返回值必须一样。**  
如果一个key对象在put的时候调用hashCode()决定了存放的位置，而在get的时候调用hashCode()得到了不一样的返回值，这个值映射到了一个和原来不一样的地方，那么肯定就找不到原来那个键值对了。

**2.hashCode()的返回值相等的对象不一定相等，通过hashCode()和equals()必须能唯一确定一个对象。**不相等的对象的hashCode()的结果可以相等。hashCode()在注意关注碰撞问题的时候，也要关注生成速度问题，完美hash不现实。

**3.一旦重写了equals()函数（重写equals的时候还要注意要满足自反性、对称性、传递性、一致性），就必须重写hashCode()函数。**而且hashCode()的生成哈希值的依据应该是equals()中用来比较是否相等的字段。

如果两个由equals()规定相等的对象生成的hashCode不等，对于hashMap来说，他们很可能分别映射到不同位置，没有调用equals()比较是否相等的机会，两个实际上相等的对象可能被插入不同位置，出现错误。其他一些基于哈希方法的集合类可能也会有这个问题